A Quartile-Based Hyper-Heuristic for Solving the 0/1 Knapsack Problem

Fernando Gómez-Herrera, Rodolfo A. Ramirez-Valenzuela,
José Carlos Ortiz-Bayliss, Ivan Amaya, and Hugo Terashima-Marín

Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias
gomezhyuuga@acm.org, rodolfo@ramirezvalenzuela.com,
{jcobayliss, iamaya2, terashima}@itesm.mx

Abstract. This research describes three novel heuristic-based approaches for solving the 0/1 knapsack problem. The knapsack problem, in its many variants, arises in many practical scenarios such as the selection of investment projects and budget control. As an NP-hard problem, it is not always possible to compute the optimal solution by using exact methods and, for this reason, the problem is usually solved by using heuristic-based strategies. In this document, we use information of the distributions of weight and profit of the items in the knapsack instances to design and implement new heuristic-based methods that solve those instances. The solution model proposed in this work is two-fold: the first part focuses on the generation of two new heuristics, while the second explores the combination of solving methods through a hyper-heuristic approach. The heuristics proposed, as well as the hyper-heuristic model, were tested on a heterogeneous set of knapsack problem instances and compared against four heuristics taken from the literature. One of the proposed heuristics proved to be highly competent with respect to heuristics available in the literature. By using the hyper-heuristic, a solver that dynamically selects heuristics based on the problem features, we improved the results obtained by the new heuristics proposed and, achieved the best results among all the methods tested in this investigation.

Keywords: Heuristics, Hyper-heuristics, Knapsack problem, Quartile

1 Introduction

Combinatorial optimization arises in a large number of events in real-world-problems. Among those problems, the knapsack problem (KP) (including all its variants) is one of the most challenging ones due to its relevance and impact, in both academic and industrial settings. The KP states the following: given a set of items, each with a weight and a profit, determine which items to include in a knapsack so that the weight is less or equal than a given limit, and the total profit is as large as possible. This version of the problem is usually referred to as the 0/1 KP, since the items are indivisible. The KP is classified as an NP-hard problem [10]. Then, in the worst case, this problem cannot be solved in a reasonable time.

The KP is the root of several interesting problems such as operations research and polynomial factorization. It is also widely used in daily life tasks. For instance, while a person is doing his luggage for an incoming trip, the airlines usually give a maximum capacity to carry in the flight. Then, determining the items the passenger may carry during the journey can be seen as a simple KP. Other real-world applications that may be modeled as KPs include load balancing problems [13], project selection problems [17] and capital budgeting problems [7].

The many different strategies that can be used to solve the KP are mainly classified into two large groups: exact methods and approximated ones. As the name indicates, the solutions obtained by exact methods are exact and optimal. Solutions obtained by approximated methods are usually suboptimal, as they approximate the solution by making some assumptions to simplify the solving process. Due to the computational resources and the problem complexity, exact methods have limited applications since they have proved to be inefficient, particularly for large problems.

The aim of this paper is to explore the use of quartile-based information of the distributions of weight and profit of the items in the KP instances to produce competitive heuristic-based methods that outperform other heuristics when tested on instances with different features. Two main contributions are derived from this investigation:

1. Two new heuristics that select the next item to pack by using quartile-based information of the distributions of weight and profit of the items.
2. A hyper-heuristic –a high-level heuristic– that is capable of selecting one suitable heuristic according to the current problem state. This hyper-heuristic outperforms the rest of the heuristics considered for this work (even the new ones proposed in this investigation).

The paper is organized as follows. Section 2 presents the basics of the KP and several approaches that have commonly been used solve the problem. The methodology followed throughout this investigation is described in Section 3. Section 4 carefully describes the heuristics used and the solution model proposed in this work. In Section 5 we present the experiments conducted and the results obtained, as well as their analysis and discussion. Finally, we present the conclusion and future work in Section 6.

2 Background

Solving the KP requires a technique that selects, among the many different groups of items that can be formed, the one that maximizes the sum of the profits in the knapsack without exceeding its capacity [22]. Among the many different methods to solve the KP we can mention: tabu search [1,3,6,18], scatter search [11], local search [12], and ant colony optimization [4, 8, 9].

There is always a trade-off between the quality of the solution and consumption of resources. For example, some techniques rely on auxiliary lists that

increase the use of the memory [18]. To avoid the high consumption of computational resources, heuristic-based methods are commonly used to solve the KP, since they consume fewer resources and lead to good-quality solutions faster than exact methods. Unfortunately, heuristic-based methods pay the price of simplicity by not guaranteeing that the optimal solution will always be found.

Some innovative approaches for solving the KP have been developed in recent years. For example, some probabilistic models include Cohort Intelligence (CI) [14], which is a new emerging technique inspired in human social interactions like learning from each other experiences. Because CI modeling involves dealing with the probability of constraints that change along with the distribution of the problem items, it is capable of jumping out of local minima. Another recent work for solving the KP involves Greedy Degree and Expectation Efficiency (GDEE) [15], which uses a technique similar to the work described in this document. The authors divide the items by a set of partitions: current item regions, candidate regions and unknown regions. This is done by a dynamic rearrangement of the items using a static objective function based on a greedy strategy. Their experiments are done by using fifteen instances from the Standard Test Case Libraries (STCL) and, for each solution, the best profit, worst profit and running time are provided. Hybrid heuristics have also proven to be useful for solving this problem. For example, by using Mean Field Theory [2], the authors generate a probabilistic model capable of replacing a difficult distribution of items by an easier one.

Another solving approach that has become popular in the last decade for other optimization problems is hyper-heuristics [20,21]. These high-level heuristic-based strategies provide a set of guidelines or strategies to develop heuristic optimization. Some authors describe hyper-heuristics simply as "heuristics to choose heuristics". Hyper-heuristics [23], particularly selection ones, are high-level heuristics that control a set of low-level heuristics and decide when and where to apply each low-level heuristic based on the problem state at the moment of the decision.

Most of the recent work on hyper-heuristics has focused on two hyper-heuristic approaches [5]: selection hyper-heuristics and generation ones. Selection hyper-heuristics generate a strategy that chooses, among a set of available heuristics, the one that best suits the current problem characterization. Solving a problem by using a hyper-heuristic implies applying one heuristic at the time, based on the features of the problem state. By applying such a heuristic, the problem state will change, and then the process will be repeated until the problem is solved.

3 Methodology

The process followed in this research is divided into three steps: data collection, model definition and experimentation.

Data collection. A total of 400 KP instances were used to test the heuristic-based methods considered for this investigation. All the instances were solved

by each of the methods. More details on the instances considered for this investigation can be found in Section. 5.

Model definition. The heuristic methods proposed in this investigation use quartile-based information to select the next item to pack. By using the information from the quartiles, we propose to split the items into three groups. For example, if we consider the weight, the sets that can be formed are $Q1_W$ (which contains the lightest 25% of the items), $Q4_W$ (which contains the heaviest 25% of the items), and IQR_W (which contains the remaining 50% of the items). Three sets of items can be generated in a similar way by considering the profit of the items: $Q1_P$, IQR_P and $Q4_P$. A graphical example of this division of the items is depicted in Figure 1. By using this intuitive conception of rules, we proposed two heuristics for solving the KP: QBH-01 and QBH-02 (both heuristics will be explained in Section 4.1). This investigation goes beyond the proposal of two new heuristics, as it also proposes a third heuristic-based approach: a hyper-heuristic (QBHH) that selectively applies QBH-01 and QBH-02, as well as one usually competent heuristic already defined in the literature, max profit.

Experimentation. In this stage, we tested our heuristic-based methods on each of the 400 instances. Related to the hyper-heuristic model, we also explored the arrangement of the selection rules in order to find which one of the new heuristics should take priority on the decision process. Two different metrics were used to evaluate the performance of the methods considered for this investigation: the local win rate (LWR) and the global win rate (GWR). LWR is a tournament-based-metric that initializes a counter of 'wins' to zero. Then, each test instance is solved and the method that obtains the maximum profit for that instance (considering only the results of the other available solvers), increases its counter by one. Thus, this metric records the fraction of instances where a method obtains the best results compared to the other methods tested. It is worth mentioning that, in some cases, two or more methods may obtain the best profit. In such cases, the counters of both methods are increased, as they both obtained the best result. GWR extends LWR by incorporating the profit of the optimal solution into the calculation. Just as in LWR, GWR uses the number of 'wins', but in a completely different fashion. It uses a threshold as a parameter and also the profit of the optimal solution[1]. The concept of winning for this metric is related to how close a solution is to the optimal one. Then, a method is the winner of a given instance if $P_h \geq \theta \times P^*$, where P_h is the profit of the solution produced by the method being evaluated, $\theta \in [0, 1]$ is a threshold value, and P^* is the profit of the optimal solution. For example, $\theta = 0.99$ means that the method wins only if it obtains a profit at least large as 99%

[1] As the reader may have already noticed, GWR requires to know the optimal solution for the instances used for testing the methods. Given the sizes of the instances studied in this work, the optimal solution was found through dynamic programming. Unfortunately, this metric might not be useful for other instances, as for some of them it could be unfeasible to obtain the optimal solution.

of P^*. By setting $\theta = 1$ we consider winners only the methods that obtain the optimal solution. For this work, we used two values for θ: $\theta = 0.99$ and $\theta = 1$.

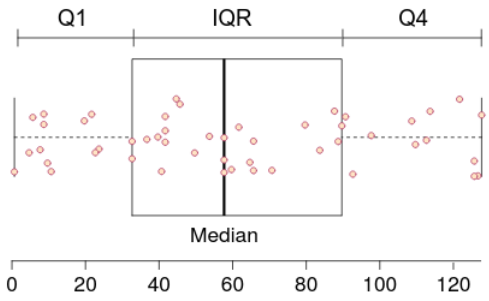

Fig. 1. Boxplot showing the sets constructed by using the distribution of the items according to the quartiles.

4 Solution model

This section describes the heuristics considered for this investigation, how they work and how they are combined into a selection hyper-heuristic.

4.1 Heuristics

All the heuristics considered for this investigation work as follows. They pack one item at the time, according to one particular criterion. The process is repeated until no more items can be packed into the knapsack. Because the criteria are different, they give place to many different strategies. For example:

Default. Packs the next item that fits within the current capacity of the knapsack.
Max profit. Packs the item with the highest profit.
Max profit per weight Packs the item with the highest ratio of profit over weight.
Min weight. Packs the item with the minimum weight.

Please note that for this work, any item that exceeds the capacity of the knapsack is removed from the list of available items. This removal procedure is invoked every time a heuristic is to be applied. Thus, the heuristics only work with items that fit into the knapsack.

4.2 Quartile-Based Heuristics

This work not only relies on existing heuristics, but proposes two new ones which are based on a combination of previously defined criteria.

QBH-01. This heuristic selects first the item with the highest profit from 'light' items (the item with the highest profit in $Q1_W$). It then selects the item from those whose weight is not "very low" nor "very high" (items in IQR_W). Finally, QBH-01 selects, from these two items, the one with the largest profit and packs it into the knapsack.

QBH-02. This heuristic follows the rationale "if there is an item with large profit and it is not too heavy, then take it". Thus, this heuristic selects the item that maximizes the profit among all the items with profits larger than $\bar{p} + \sigma_p$ and weights in IQR_W. Where $\bar{p}$ is the mean of the profit distribution and σ_P is the standard deviation of the profit distribution.

4.3 A Quartile-based Hyper-heuristic

In this section, we describe a novel selection hyper-heuristic approach for solving the KP. The general idea is to define a high-level heuristic that controls single heuristics such as QBH-01, QBH-02 and max profit. The core of the hyper-heuristic model, the module that selects which heuristic to apply, uses a fixed sequence of heuristics, which are applied only when certain conditions are satisfied.

Figure 2 presents the pseudo-code for QBHH. In a graphical way, the behaviour of the hyper-heuristic is depicted in Figure 3.

The hyper-heuristic proposed in this investigation was obtained empirically, based on the observations of the solving methods and their performance on the instances used in this work. As the reader may observe, QBHH mainly relies on the two quartile-based heuristics proposed in this investigation, QBH-01 and QBH-02; and only when these heuristics fail to select an item, based on their particular criteria, max profit is used.

To clarify the process followed by the hyper-heuristic when a KP instance is solved, Figure 4 presents an example of the iterations and how the distributions of weight and profit of the items change throughout the solving process.

5 Experiments and Results

We trained and tested the heuristic-based methods on a set of KP instances with different features. These instances are grouped into four groups of 100 instances each. The instances are balanced with respect to four existing heuristics, so the arrangement of the items inside each group is designed to favour different heuristics. Each instance considered for this investigation contains 50 items with a capacity of 50 units of weight. The items have at most 20 units of weight. The profits of the items range from 1 and 128. The instances used in this investigation are available at `http://bit.ly/KnapsackInstances`.

```
Knapsack ← ∅
Items ← {I_1, I_2, ... I_n}
while (Knapsack.Weight < Capacity) and (Items.Size > 0) do
    for each item ∈ Items do
        if item.Weight + Knapsack.Weight > Capacity then
            Items.Remove(item)
        end if
    end for
    x ← max(item.Profit) ∈ Q1_W
    y ← max(item.Profit / item.Weight) ∈ IQR_W
    if x and y then
        Knapsack.Add(max(x.Profit, y.Profit))                    ▷ QBH-01
    end if
    x ← item ∈ Items | item.Profit > σ_p + p̄ and item.Weight ∈ IQR_W
    if x then
        Knapsack.Add(x)                                          ▷ QBH-02
    end if
    x ← max(item.Profit) ∈ Items
    Knapsack.Add(x)                                              ▷ Max profit
end while
```

Fig. 2. An algorithmic description of QBHH. Comments on the right indicate the heuristic used by each rule.

Table 1. Comparison of the different methods considered for this investigation by using the metrics LWR, GWR ($\theta = 0.99$) and GWR ($\theta = 1$). The best performer for each metric is highlighted in bold.

Method	**LWR**	**GWR** ($\theta = 0.99$)	**GWR** ($\theta = 1$)
Default	14.25%	9.50%	5.75%
Max profit	24.00%	24.25%	22.25%
Max profit per weight	24.25%	25%	19.75%
Min weight	23.75%	21.00%	12.50%
QBH-01	45.75%	44.25%	35.50%
QBH-02	20.25%	21.25%	18.00%
QBHH	**49.50%**	**49.00%**	**38.25%**

Table 1 shows the performance of the different heuristic-based methods studied in this investigation by using the three metrics previously described. From the simple heuristics, QBH-01 is clearly the best option, as it obtains the best results by using the three metrics. The second quartile-based heuristic proposed in this investigation, QBH-02, performed poorly on this set of instances. By combining the quartile-based heuristics into a hyper-heuristic we obtained the best performer of the methods in this investigation: QBHH. When the metric LWR is considered, the hyper-heuristic wins in almost 50% of the instances, which is more than twice the best heuristic taken from the literature, max profit per weight. The performance of QBHH represents a small improvement in perfor-

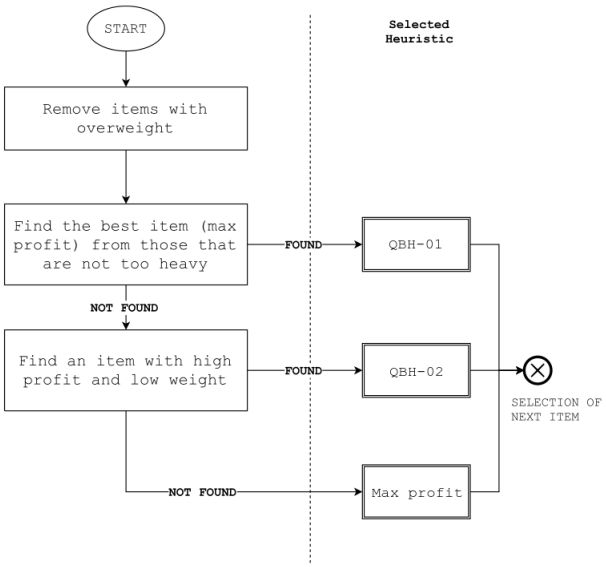

Fig. 3. High-level description of QBHH. The process depicted in this figure is repeated until no more items can be packed into the knapsack. Every time an item is selected, the process starts over and, by doing so, the attributes of the problem change as well as the sets that group the items by using the quartiles. Whenever QBH-01 and QBH-02 fail to find an item to pack based on their respective criteria, they select the next item to pack according to the max profit heuristic.

mance with respect to QBH-01, but a large one for any of the other heuristics (around 20% for any of the metrics). The fact that QBHH relies mainly on QBH-01 explains this behaviour, as the first rule tries to apply QBH-01 and, only if it is unable to find an item, it tries QBH-02 –which proved a poor performer when applied in isolation.

A deeper analysis on the process conducted by QBHH shows that the hyper-heuristic applies QBH-01 to pack around 86% of the items in the instance set. The proportion of items that are packed by using QBH-02 and max profit drops to 6.76% and 7.46%, respectively.

5.1 Discussion

Does the order of the heuristics in the hyper-heuristic process affect the performance of the hyper-heuristic? Aiming at answering this question we inverted the order of QBH-01 and QBH-02 in QBHH. By changing the order of the heuristics (so that the first available heuristic was QBH-02 instead of QBH-01) the performance of the hyper-heuristic decreased by half. This behaviour is interesting, as the same heuristic, used at a different position of the sequence, leads to very different results.

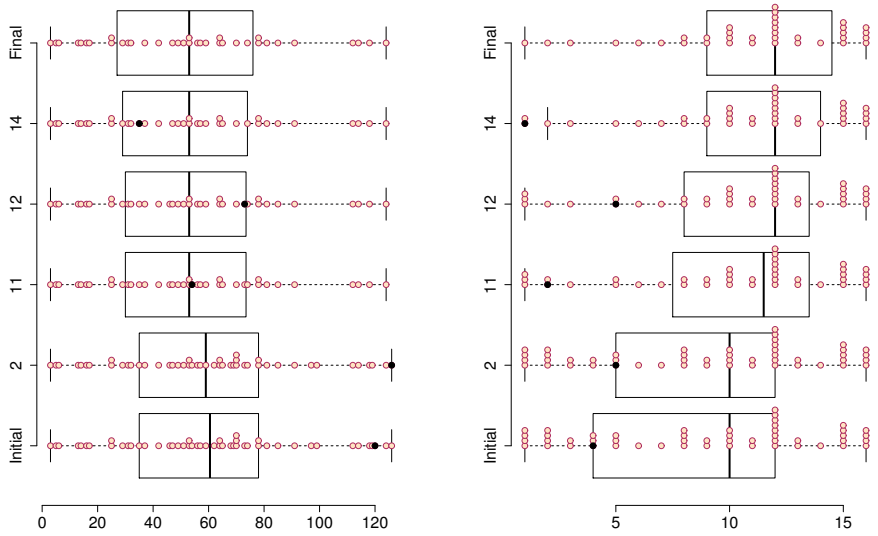

Fig. 4. Example of how the distribution of weight and profit of the items changes as the solving process takes place by using QBHH. Left and right figures represent the profit and weight distributions at different iterations of the solving process, respectively. Black circles indicate the selected item for that iteration. The left axis labels indicate the current iteration of the selection process. The 'Initial' and 'Final' labels indicate the status of the distribution at the initial state and at the end of the solving process, respectively.

As part of a further analysis, we replaced QBH-02 by max profit per weight, and use it as a second option in the sequence of available heuristics for the hyper-heuristic. The rationale for this change was to explore how the second best of the heuristics could fit into the hyper-heuristic model. Contrary to what we expected, the performance was significantly below the one shown by the original sequence QBH-01/QBH-02/max profit.

6 Conclusion and Future Work

This work describes two new heuristics for solving the KP. These heuristics use the information of the distributions of weight and profit of the items in the problem to decide which item to pack next. Specifically, these new heuristics (QBH-01 and QBH-02) split the items into three groups by feature, based on the quartile-information of the distributions. Although the idea seems simple, the results proved that one of these heuristics, QBH-01, outperformed four heuristics

taken from the current literature in any of the three metrics considered for this work. Regarding QBH-02, its performance (when applied in isolation) was below our expectations and their results were rather poor.

Trying to improve the performance of the solving methods, we intuitively proposed a way to combine QBH-01, QBH-02 and max-profit to further improve the solution process. The hyper-heuristic obtained outstanding results, outperforming all the other heuristics –even QBH-01.

There are various interesting paths which may be worth exploring in the future. The first one is related to how the quartile-based heuristics are defined. For example, the boundaries for selection might be automatically generated by using genetic programming or a similar approach. The hyper-heuristic itself is another field of opportunity. For this work, we proposed the sequence of heuristics by empirical reasons but various methods for producing hyper-heuristics have been proposed in the last years [16, 19]. It will be an interesting future work to use one or more of such models to define the sequence of heuristics to apply.

Acknowledgments

This research was supported in part by CONACyT Basic Science Projects under grant 241461 and ITESM Research Group with Strategic Focus in intelligent Systems.

References

1. A. Amuthan and K. Deepa Thilak. Survey on tabu search meta-heuristic optimization. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pages 1539–1543, Oct 2016.
2. J. Banda, J. Velasco, and A. Berrones. A hybrid heuristic algorithm based on mean-field theory with a simple local search for the quadratic knapsack problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2559–2565, June 2017.
3. V. Barichard and J. K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Tsinghua Science and Technology*, 8(1):8–13, Feb 2003.
4. Edmund Burke and Graham Kendall. *Search methodologies : introductory tutorials in optimization and decision support techniques.* Springer, 2005.
5. Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. *A Classification of Hyper-heuristic Approaches*, pages 449–468. Springer US, Boston, MA, 2010.
6. Y. H. Chou, Y. J. Yang, and C. H. Chiu. Classical and quantum-inspired tabu search for solving 0/1 knapsack problem. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1364–1369, Oct 2011.
7. Xiaoling Cui, Dazhi Wang, and Yang Yan. Aes algorithm for dynamic knapsack problems in capital budgeting. In *2010 Chinese Control and Decision Conference*, pages 481–485, May 2010.
8. Marco Dorigo and Thomas Sttzle. *The Ant Colony Optimization Metaheuristic*, pages 25–64. MIT Press, 2004.

9. E. O. Gagliardi, M. G. Dorzn, M. G. Leguizamn, and G. H. Pealver. Approximations on minimum weight pseudo-triangulation problem using ant colony optimization metaheuristic. In *2011 30th International Conference of the Chilean Computer Science Society*, pages 238–246, Nov 2011.
10. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
11. M. Hifi and N. Otmani. A first level scatter search for disjunctively constrained knapsack problems. In *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, pages 1–6, March 2011.
12. A. Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, Aug 2002.
13. Z. I. Kiss, A. C. Hosu, M. Varga, and Z. A. Polgar. Load balancing solution for heterogeneous wireless networks based on the knapsack problem. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–6, July 2015.
14. Anand J Kulkarni and Hinna Shabir. Solving 01 Knapsack Problem using Cohort Intelligence Algorithm. *Int. J. Mach. Learn. Cybern.*, 7(3):427–441, 2016.
15. Jianhui Lv, Xingwei Wang, Min Huang, Hui Cheng, and Fuliang Li. Solving 0-1 knapsack problem by greedy degree and expectation efficiency. *Appl. Soft Comput. J.*, 41:94–103, 2016.
16. Mashael Maashi, Ender zcan, and Graham Kendall. A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, 41(9):4475 – 4493, 2014.
17. M. Naldi, G. Nicosia, A. Pacifici, U. Pferschy, and B. Leder. A simulation study of fairness-profit trade-off in project selection based on hhi and knapsack models. In *2016 European Modelling Symposium (EMS)*, pages 85–90, Nov 2016.
18. S. Niar and A. Freville. A parallel tabu search algorithm for the 0-1 multidimensional knapsack problem. In *Proceedings 11th International Parallel Processing Symposium*, pages 512–516, Apr 1997.
19. José Carlos Ortiz-Bayliss, Hugo Terashima-Marín, and Santiago Enrique Conant-Pablos. Combine and conquer: an evolutionary hyper-heuristic approach for solving constraint satisfaction problems. *Artificial Intelligence Review*, 46(3):327–349, Oct 2016.
20. Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. A comprehensive analysis of hyper-heuristics. *Intell. Data Anal.*, 12(1):3–23, January 2008.
21. Z. Ren, H. Jiang, J. Xuan, Y. Hu, and Z. Luo. New insights into diversification of hyper-heuristics. *IEEE Transactions on Cybernetics*, 44(10):1747–1761, Oct 2014.
22. D. Sapra, R. Sharma, and A. P. Agarwal. Comparative study of metaheuristic algorithms using knapsack problem. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 134–137, Jan 2017.
23. Hugo Terashima-Marín, EJ Flores-Alvarez, and Peter Ross. Hyper-heuristics and classifier systems for solving 2d-regular cutting stock problems. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 637–643. ACM, 2005.